키 공유 확인이 가능한 새로운 키 합의 프로토콜

송보연, 김광조
한국정보통신대학원대학교

2-pass Unilateral Authenticated Key agreement protocol with Key confirmation

Boyeon Song, Kwangjo Kim
Information and Communications University

요 약

본 논문에서는 ANSI, IEEE, ISO/IEC, NIST 에서 표준화되거나 표준화 과정 중에 있는 KEA, Unified Model, MQV 와 같은 인증된 키 합의 프로토콜들을 고찰하고, 이를 보안한 새로운 인증된 키 합의 프로토콜을 생성한다. 이로부터 키 소유 확인 기능을 더한 키 합의 프로토콜을 설계하고 이러한 키 합의 프로토콜이 6 가지 요구되는 안전성을 만족함을 증명한다. 또, 설계한 프로토콜에서 메시지 수를 줄이고 상호 실체 인증과 키 인증 및 일방향 키 소유 확인이 가능한 2-pass 키 합의 프로토콜을 제안한다.

1. 서론

일반적으로 둘 이상의 실체는 암호 통신을 위하여 사전에 공개된 통신로를 통하여 키를 공유하여야 하는 키 확립(key establishment) 과정을 필요로 한다. 공유된 키는 메시지 기밀성, 무결성, 개인 식별 등과 같은 암호학적 서비스를 달성하기 위해 사용되며, 이상적으로는 이렇게 확립된 키는 직접 만나서 키를 교환한 것과 같은 특성[1]을 가져야 한다. 키 확립 프로토콜은 크게 키 전송(key transport)과 키 합의(key agreement)로 나누어 생각할 수 있다. 키 전송 프로토콜은 한 쪽 실체가 키를 생성하여 다른 실체에게 안전하게 전달하는 프로토콜이며, 키 합의 프로토콜은 양쪽 실체가 상호작용을 통하여 서로 주고 받은 정보를 사용해서 공유키를 생성하는 프로토콜이다[8].

키 확립을 위해 대칭키 암호 시스템을 사용할 경우 사용자의 수(n)가 늘어날 수록 공유하고 관리해야 하는 키의 숫자가 약 사용자 수의 제곱($_nC_2$)에 비례하여 크게 증가하므로 키 관

리상의 어려운 점이 있다. 반면, 공개키 암호 시스템을 이용하면 각 사용자는 자신의 개인키만을 안전하게 보관하면 되고 전자서명과 같은 방법으로 메시지의 인증성을 제공할 수 있는 장점이 있다[1,4,13].

지금까지 키 합의 및 인증 프로토콜로서 다양한 프로토콜들이 제안되었으며 그 중 많은 프로토콜들이 여러 가지 공격 방법들에 대해 안전하지 못하다는 것이 증명되었다. 따라서 어떤 프로토콜을 분석하기 전에 프로토콜이 저항해야 하는 공격은 어떤 것들이 있고, 어떤 속성을 만족해야 하는지 정의할 필요가 있다.

키 합의 프로토콜에 대한 공격 모델로서는 수동 공격(passive attack)과 능동 공격(active attack)이 있다. 수동 공격은 합법적인 실체들이 프로토콜을 수행하는 것을 공격자가 단지 관찰하는 것만으로 프로토콜의 목적을 달성하는 것을 방해하는 것이며, 능동 공격은 공격자가 통신 내용을 삭제, 수정, 삽입 및 되풀이(replay) 등의 다양한 방법으로 통신 내용을 파괴하는 것이다. 키 합의 프로토콜은 공개된 네트웍에서 수행하므로 안전한 프로토콜이 되기 위해서는 이러한 모든 공격모델에 대응해서 안전하게 설계되어야 한다[1,8].

본 논문은 공개키 암호 시스템을 이용하여 두 실체로 구성된 인증된 키 합의 프로토콜에 대해 설계하고자 한다. 키 합의 프로토콜의 안전성은 Diffie-Hellman 문제의 어려움에 기반하고 있다. 제안한 2-pass AK(Authenticated Key agreement) 프로토콜은 Diffie-Hellman 기반의 키 합의 프로토콜[17]로서 [1]에서 논의하는 많은 안전성 및 성능 요구 사항을 만족한다. 이 프로토콜로부터 2 가지 변형된 프로토콜을 제시한다. 먼저 프로토콜에 메시지 인증 알고리즘(Message Authentication Code)을 추가하여 3-pass AKC(Authenticated Key agreement with key Confirmation) 프로토콜을 생성한다. 다음으로 AKC 프로토콜과 거의 같은 안전성을 제공하면서 2-pass 인 일방향 AKC 프로토콜을 설계한다. 서명값을 추가하기 때문에 계산량은 보다 많지만 메시지 수가 줄기 때문에 성능면에서 3-pass AKC 보다 효율적이다.

논문의 구성은 다음과 같다. 2 장에서는 키 합의 프로토콜의 요구사항을 요약해서 기술한다. 3 장에서는 ANSI, IEEE, ISO/IEC, NIST 에서 표준화되거나 표준화 과정 중에 있는 KEA 와 Unified Model 과 MQV 프로토콜에 대해 개략적으로 살펴보고 AKC 프로토콜에 대해 설명한다. 4 장에서는 3 가지 새로운 키 합의 프로토콜들을 제안하다. 5 장에서는 기존의 프로토콜과 제안한 프로토콜을 비교 분석한다. 6 장에서는 본 연구에 대해 결론을 맺고 향후과제를 제시한다.

2. 키 합의 프로토콜의 요구사항

키 합의 프로토콜의 궁극적인 목표는 키로 사용할 비밀 정보를 안전하게 공유하는 것이며 이상적으로는 직접 키를 합의한 것과 같은 특성을 가지도록 하는 것이다. 즉, 키를 생성하기 위한 비밀 정보는 합법적인 객체들 간에만 공유되어야 하고, 키는 키 생성이 가능한 공간에서 난수적으로 선택되며, 불법적인 객체는 키에 대한 어떠한 부분 정보도 얻을 수 없어야 한다. 이 장에서는 키 합의 프로토콜의 요구사항을 기본적 안전성 요구사항, 추가적 안전성 요구사항, 성능 요구사항으로 구분하여 기술한다[2,3,8].

가. 안전성 요구사항

1) 기본적 안전성 요구사항

기본적 안전성 요구사항은 키 합의 프로토콜이 반드시 제공해야 하는 기본적인 보안을 위한 요구사항을 말한다.

A 와 B 는 프로토콜을 바르게 수행하는 합법적인 두 실체라고 본다.

- **함축적 키 인증성 (IKA, Implicit Key Authentication)**

  실체 A 가 실체 B 이외에 어느 누구도 공유키를 생성할 수 없다는 확신을 가질 수 있을 경우 키 합의 프로토콜은 A 에게 B 에 대한 함축적 키 인증성(IKA)을 제공한다고 말한다.

  A 에게 B 에 대한 IKA 를 제공하고 B 에게 A 에 대한 IKA 를 제공하는 키 합의 프로토콜을 ***AK(Authenticated Key agreement) 프로토콜***이라고 한다.

- **명시적 키 인증성 (EKA, Explicit Key Authentication)**

  실체 B 가 실제로 공유키를 계산하여 소유하는 것을 실체 A 가 확신할 수 있을 경우 프로토콜은 A 에게 B 에 대한 명시적 키 인증성(EKA)을 제공한다고 말한다. EKA 는 IKA 에 다른 실체가 키를 가지고 있음을 확신하는 ***키 확인(Key Confirmation)*** 기능을 더한 것이다.

  A 에게 B 에 대한 EKA 를 제공하고 B 에게 A 에 대한 EKA 를 제공하는 키 합의 프로토콜을 ***AKC(Authenticated Key agreement with key Confirmation) 프로토콜***이라고 한다.

2) 추가적 안전성 요구사항

이것은 응용분야에 따라서 추가적으로 요구되는 안전성 요구사항을 말한다.

- **알려진 키에 대한 안전성 (K-KS, Known-Key security)**

  A 와 B 사이의 키 합의 프로토콜의 각 세션(session)에서는 유일한 비밀키를 생성해야 하는데 이때 생성한 비밀키를 ***세션키***라 부른다. 이전의 다른 세션키들이 공격자에게 노출되었을 경우에도 프로토콜의 안전성이 보장되는 것을 말한다.

- **전향적 보안성 (FS, Forward Secrecy)**

  하나 또는 둘 이상의 실체들의 장기적인 개인키(long-term private key)가 노출되었을 경우에도 이전에 합의한 세션키들에 대한 안전성이 제공되는 경우를 말한다.

- **키 위장(K-CI, Key-Compromise Impersonation)에 대한 안전성**

  실체 A 의 장기적인 개인키가 공격자에게 노출되었을 경우 공격자는 A 로 당연히 위장할 수 있다. 그러나 공격자는 A 에게 다른 실체 B 인 것처럼 위장할 수 없는 안전성을 말한다.

- **미지의 키 공유(UK-S, Unknown Key-Share)에 대한 안전성**

  실체 A 는 실체 B 와 키를 공유하고 있다고 믿고 있지만, B 는 A 와 키를 공유하고 있

는 것이 아니라 공격자 E 와 공유하고 있다고 믿게 하는 공격을 미지의 키 공유 공격(Unknown Key-Share Attack)이라고 한다. 이 공격에 대해 안전한 경우를 말한다.

본 논문에서는 위의 안전성 요구사항을 IKA, EKA, K-KS, FS, K-CI, UK-S 로 표시한다.

나. 성능 요구사항

- 메시지 교환 횟수를 최소화한다.
- 전송되는 비트 수를 작게 한다.
- 산술적 계산량을 줄인다.
- 온라인 계산량이 감소하도록 사전계산을 가능하게 한다.

3. 최근 표준안에 제안된 키 합의 프로토콜

이 장에서는 ANSI, IEEE, ISO/IEC, NIST 에 의해 현재 표준화되었거나 표준화되는 과정 중에 있는 키 합의 프로토콜인 KEA, Unified Model, MQV 에 대해 개략적으로 설명한다. 이 프로토콜들의 안전성은 Diffie-Hellman 문제의 어려움에 기반을 두고 있다.

Diffie-Hellman 문제는 위수가 소수 n 인 순환군 G 에서 생성자 g 에 대하여 원소 g^x, g^y $(x, y \in_R [1, n-1])$ 가 주어졌을 때 g^{xy} 를 발견하는 문제이다. 이 문제는 널리 연구된 이산 대수 문제와 밀접한 관련이 있다. 이산 대수 문제는 위수가 소수 n 인 순환군 G 에서 생성자 g 에 대하여 g^x $(x \in_R [1, n-1])$ 가 주어졌을 때 x 를 발견하는 문제이다. 이 논문에서는 소수 p 를 모듈로하는 정수의 곱셈군(Z_p^*)의 부분군이면서 소수를 위수로 하는 군 G 상에서 생각하도록 한다. 여기에서 논의되는 사항은 이산 대수 문제가 계산적으로 다루기 어려우면서 소수를 위수로 하는 어떤 군에서도 동등하게 적용할 수 있다. 예를 들면, 소수를 위수로 하는 유한체 상의 타원곡선에 있는 점들로 이루어진 군의 부분군에 대해서도 성립한다.

다음의 기호들을 이 논문 전체에 걸쳐 공통적으로 사용한다.

A, B	정직한 실체들
p	1,024 비트의 소수
q	$p-1$ 을 나누는 160 비트의 소수
g	Z_p^* 에서 위수가 q 인 군의 생성자
a, b	A, B 의 장기 개인키; $a, b \in_R [1, q-1]$
Y_A, Y_B	A, B 의 장기 공개키; $Y_A = g^a \bmod p,\ Y_B = g^b \bmod p$
x, y	A, B 의 임시 개인키; $x, y \in_R [1, q-1]$
R_A, R_B	A, B 의 임시 공개키; $R_A = g^x \bmod p,\ R_B = g^y \bmod p$
H	암호학적 해쉬함수 (예, SHA-1)

$(p,\ q,\ g)$ 는 모든 실체들에게 공통적으로 알려져 있고 이후 프로토콜에서 장기 공개키는 공개키 증명서에 의해 교환한다고 가정한다. $Cert_A$ 는 A 임을 증명하는 ID 정보, A 의 장기 공개키 Y_A 와 이 정보에 대한 신뢰할 만한 서명기관(CA)의 서명값으로 구성된 공개키 증명서이다.

CA 는 A 가 장기 공개키 Y_A 에 대응하는 비밀키 a 를 가지고 있음을 검증할 수 있다고 가정한다. 이것은 공격자 E 가 A 의 공개키 Y_A 를 자신의 것으로 등록하고 A 의 메시지가 E 로부터 온 것이라고 B 를 속이는 UK-S 공격을 막기 위함이다.

또한 CA 가 A 의 장기 공개키 Y_A 의 타당성을 검증했다고 가정한다. 이는 $1 < Y_A < p$ 이고 $(Y_A)^q \equiv 1 \pmod{p}$ 인지를 확인함으로 가능하고 이를 공개키 확인(public key validation)[13] 이라 부른다.

가. KEA

KEA(Key Exchange Algorithm)은 NSA 에서 설계했고 1998 년 5 월에 기밀 분류에서 해제됐다. 이 프로토콜은 1994 년에 NSA 가 설계한 키 위탁 기능을 제공하는 FORTEZZA 카드에서 사용하는 키 합의 프로토콜이다[10].

- **[프로토콜 1]**

① A 와 B 는 각각 상대방의 신뢰할 만한 공개키 Y_A 와 Y_B 의 사본을 얻는다.
② A 는 $x \in_R [1, q-1]$ 를 선택하고 B 에게 $R_A = g^x$ 를 보낸다.
③ B 는 $y \in_R [1, q-1]$ 를 선택하고 A 에게 $R_B = g^y$ 를 보낸다.
④ A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod{p}$ 인지를 검증한다.
만약 검사가 실패하면, A 는 프로토콜을 실패로 종료한다.
그렇지 않으면, A 는 공유 비밀키 $K = (Y_B)^x + (R_B)^a \bmod p$ 를 계산한다.
만약 $K = 0$ 이면, A 는 프로토콜을 실패로 종료한다.
⑤ B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod{p}$ 인지를 검증한다.
만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다.
그렇지 않으면, B 는 공유 비밀키 $K = (Y_A)^y + (R_A)^b \bmod p$ 를 계산한다.
만약 $K = 0$ 이면, B 는 프로토콜을 실패로 종료한다.
⑥ A, B 둘 다는 80 비트 세션키 $k = kdf(K)$ 를 계산한다.

이 프로토콜에서는 임시 공개키들을 교환한 후 세션키를 생성한다. 실체 A 와 B 는 $K = g^{ay} + g^{bx}$ 라는 비밀 정보를 공유하게 된다. Goss[14]와 MTI/A0[15]의 프로토콜과 유사하다. 하나의 특징으로서 kdf 라는 키 유도 함수를 사용하여 공유 비밀 정보 K 로부터 세션키 k 를 계산하도록 하였다. 이것은 공유 비밀 정보와 세션키 간의 수학적 구조를 제거함으로써

알려진 키 공격(known-key attack)을 방지하고자 함이다.

KEA 는 공격자가 두 실체 A 와 B 의 장기 개인키 a,b 를 안다면 이전의 모든 세션키들을 계산할 수 있는 약점을 가지고 있다. 즉, FS 를 제공하지 않는다.

나. Unified Model

Ankney, Johnson, Matyas[9]가 제안했고 ANSI X9.42[4], ANSI X9.63[5], IEEE P1363[6]에 표준안으로 제출되어 있는 AK 프로토콜이다. 이 프로토콜의 장점 중 하나는 개념적으로 단순해서 분석하기가 쉽다는 것이다[9].

- **[프로토콜 2]**

① A 는 $x \in_R [1, q-1]$ 를 선택하고 B 에게 $R_A = g^x$ 와 $Cert_A$ 를 보낸다.

② B 는 $y \in_R [1, q-1]$ 를 선택하고 A 에게 $R_B = g^y$ 와 $Cert_B$ 를 보낸다.

③ A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod p$ 인지를 검증한다.

만약 검사가 실패하면, A 는 프로토콜을 실패로 종결한다.

그렇지 않으면, A 는 비밀 정보 $K = (Y_B)^a \| (R_B)^x \pmod p$ 를 계산한다.

④ B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod p$ 인지를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종결한다.

그렇지 않으면, B 는 비밀 정보 $K = (Y_A)^b \| (R_A)^y \pmod p$ 를 계산한다.

⑤ 세션키는 $k = H(K)$ 이다.

이 프로토콜은 KEA 와 비교할 때 임시 공개키와 장기 공개키를 결합하는 대신 g^{ab}, g^{xy} 를 생성해서 해쉬함수 H 를 이용하여 세션키를 생성한다. 실체 A 와 B 는 $K = g^{ab} \| g^{xy}$ 라는 비밀 정보를 공유하게 된다.

이 프로토콜은 K-CI 안전성을 제공하지 않는다. 공격자 E 가 a 를 알게 되면 b 를 알지 않고도 B 인 것처럼 y 를 생성하고, g^{ab} 는 $(Y_B)^a$ 를 통해 g^{xy} 는 $(R_A)^y$ 로 계산해서 K 를 알 수 있기 때문이다.

다. MQV 프로토콜

MQV[2]로 불리는 이 프로토콜은 1998 년 Menezes, Qu, Vanstone 등에 의해 제안되었으며 ANSI X9.42, ANSI X9.63, IEEE P1363 에 표준안으로 제출되어 있다. 여기에서 사용되는 기호 $\overline{X}$ 는 입력 X 의 하위 비트만을 선택하는 의미를 나타내는 것으로 만약 $X \in [1, p-1]$ 라면

$\overline{X} = X \bmod 2^{80} + 2^{80}$ 이 된다. 보다 일반화하면 $\overline{X} = X \bmod 2^{f/2} + 2^{f/2}$ 이고, 이때 f 는 q 의 비트 길이이다. $\overline{X} \bmod q \neq 0$ 임을 주의하자. 이것은 승산할 때 절반 크기의 숫자만을 이용해서 효율성을 높이기 위해 사용한다.

- **[프로토콜 3]**

① A 는 $x \in_R [1, q-1]$ 를 선택하고 B 에게 $R_A = g^x$ 와 $Cert_A$ 를 보낸다.

② B 는 $y \in_R [1, q-1]$ 를 선택하고 A 에게 $R_B = g^y$ 와 $Cert_B$ 를 보낸다.

③ A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod{p}$ 인지를 검증한다.

만약 검사가 실패하면, A 는 프로토콜을 실패로 종료한다. 그렇지 않으면,

A 는 $s_A = (x + a\overline{R_A}) \bmod q$ 와 비밀 정보 $K = (R_B (Y_B)^{\overline{R_B}})^{s_A} \bmod p$ 를 계산한다.

만약 $K = 1$ 이면, A 는 프로토콜을 실패로 종료한다.

④ B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod{p}$ 인지를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다. 그렇지 않으면,

B 는 $s_B = (y + b\overline{R_B}) \bmod q$ 와 비밀 정보 $K = (R_A (Y_A)^{\overline{R_A}})^{s_B} \bmod p$ 를 계산한다.

만약 $K = 1$ 이면, B 는 프로토콜을 실패로 종료한다.

⑤ 세션키는 $k = H(K)$ 이다.

실체 A 와 B 는 $K = g^{(x+a\overline{g^x})(y+b\overline{g^y})}$ 라는 비밀 정보를 공유하게 된다.

이 프로토콜은 UK-S 공격이 가능하다는 것을 Kaliski 가 최근에 제시했다[12]. 공격자 E 는 A 의 임시 공개키 R_A 를 가로채어 $R_E = R_A (Y_A)^{\overline{R_A}} g^{-1}$ 와 장기 비밀키 $e = (\overline{R_A})^{-1} \bmod q$, 장기 공개키 $Y_E = g^e \bmod p$ 를 계산하고 CA 로부터 Y_E 에 대한 인증을 받아서 B 에게 R_E 와 $Cert_E$ 를 전송한다. 이와 같은 방법으로 B 는 A 와 같은 키를 계산하지만 E 와 그 키를 공유한다고 생각할 수 있다.

라. 인증 및 키 확인된 키 합의 프로토콜 (AKC protocol)

AKC 프로토콜은 AK 프로토콜로부터 생성된다. 다음에 나오는 3-pass AKC 프로토콜[3]은 위의 [프로토콜 2](Unified Model)에 메시지 번호, 실체의 ID, 임시 공개키를 포함하는 메시지 인증코드(MAC)를 부가함으로 생성할 수 있다. 즉, MAC 을 검증함으로써 세션키를 실제로 계산하였다는 것을 증명할 수 있게 된다. AKC 프로토콜은 결과적으로 3-pass 로 구성되고 여기에서 사용하는 H_1, H_2 는 서로 독립적인 해쉬함수이다. 실질적으로 $H_1(m) = H(10, m)$ 과

$H_2(m) = H(01, m)$ 을 선택하기도 한다. 공유키 k' 과 세션키 k 는 구분하여 사용한다. 공유키 k' 은 해쉬함수 H_1 으로 생성해서 MAC 을 계산하는 비밀키로 사용하고, 세션키 k 는 해쉬함수 H_2 로 생성해서 향후 교환되는 메시지의 기밀성을 제공하기 위해 사용한다[3,5].

- **[프로토콜 4]**

① A 는 $x \in_R [1, q-1]$ 를 선택하고 B 에게 $R_A = g^x$ 와 $Cert_A$ 를 보낸다.

② (a) B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod p$ 인지를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다.

(b) B 는 $y \in_R [1, q-1]$ 를 선택하고

$R_B = g^y$, $k' = H_1((Y_A)^b \| (R_A)^y)$, $k = H((Y_A)^b \| (R_A)^y)$ 와

$m_B = MAC_{k'}(2, B, A, R_B, R_A)$ 를 계산한다.

(c) B 는 A 에게 $R_B, Cert_B$ 와 m_B 를 보낸다.

③ (a) A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod p$ 인지를 검증한다.

만약 검사가 실패하면, A 는 프로토콜을 실패로 종료한다.

(b) A 는 $k' = H((Y_B)^a \| (R_B)^x)$, $m_B' = MAC_{k'}(2, B, A, R_B, R_A)$ 를 계산하고

$m_B' = m_B$ 인지를 검증한다.

(c) A 는 $m_A = MAC_{k'}(3, A, B, R_A, R_B)$ 와 $k = H((Y_B)^a \| (R_B)^x)$ 를 계산하고

B 에게 m_A 를 보낸다.

④ B 는 $m_A' = MAC_{k'}(3, A, B, R_A, R_B)$ 를 계산하고 $m_A' = m_A$ 인지를 검증한다.

⑤ 세션키는 k 이다.

동일한 방법으로 [프로토콜 1](KEA)과 [프로토콜 2](MQV)로부터 3-pass AKC 프로토콜을 생성할 수 있다. AKC 가 AK 보다 실제적으로 좋은 프로토콜인 이유는 키 확인 기능을 제공함으로써 좀 더 강력한 안전성 요구사항을 제공할 수 있기 때문이다. 예를 들어 위와 같은 방법으로 MQV 에 MAC 을 추가하면 UK-S 공격을 방지할 수 있다.

MAC 은 효과적으로 계산될 수 있기 때문에 AK 에 키 확인 기능을 추가하는 것은 키 확립 매커니즘에서 계산량에 큰 영향을 주지는 않는다. 그러나 메시지 수가 하나 더 증가한다.

4. 새로운 키 합의 프로토콜

이 장에서는 Diffie-Hellman 기반의 새로운 AK 프로토콜을 제안한다. 다음으로 이 프로토콜에 MAC 을 더해서 키 확인 기능을 추가한 3-pass AKC 프로토콜을 생성한다. 그리고 3-pass 와 유사한 안전성을 제공하면서 실제 프로토콜 상에서 유용하게 적용할 수 있는 2-pass 일방향 AKC 프로토콜을 설계한다.

가. AK 프로토콜

3 장에서 기술한 KEA, Unified Model, MQV 프로토콜은 각각 FS, K-CI, UK-S 에 대한 안전성을 제공하지 않는 단점을 가지고 있다.

[프로토콜 5]는 이러한 프로토콜의 단점을 고려해서 안전성을 보안하도록 설계한다.

- **[프로토콜 5]**

① A 는 $x \in_R [1, q-1]$ 를 선택하고 B 에게 $R_A = g^x$ 와 $Cert_A$ 를 보낸다.

② B 는 $y \in_R [1, q-1]$ 를 선택하고 A 에게 $R_B = g^y$ 와 $Cert_B$ 를 보낸다.

③ A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod{p}$ 인지를 검증한다.

만약 검사가 실패하면, A 는 프로토콜을 실패로 종료한다.

그렇지 않으면, A 는 공유 비밀정보 $K = Y_B{}^x \cdot R_B{}^{(a+x)} \bmod p$ 를 계산한다.

④ B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod{p}$ 인지를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다.

그렇지 않으면, B 는 공유 비밀정보 $K = Y_A{}^y \cdot R_A{}^{(b+y)} \bmod p$ 를 계산한다.

⑤ A, B 는 세션키 $k = kdf(K)$ 를 계산한다.

여기서, kdf 는 키 유도 함수로 해쉬함수나 대칭키 암호기법이 사용될 수 있다.

KEA 가 g^{ay}, g^{bx} 를 계산하고, Unified Model 이 g^{ab}, g^{xy} 를 생성해서 비밀 정보를 공유하는 반면 이 프로토콜은 g^{xy}, g^{ay}, g^{bx} 로 구성된 $K = g^{xy+ay+bx}$ 라는 공유 비밀 정보를 실체 A 와 B 가 계산하게 된다.

1) 안전성 분석

[프로토콜 5]은 그 안전성이 B-R 과 같은 분산 계산 모델[1,18]에서 형식적인 방법으로 증명된 것은 아니지만 경험적인 논증에 의해 상호 IKA 를 제공한다고 볼 수 있다. 추가적인 안전성에 대해서는 다음과 같이 설명할 수 있다.

- **알려진 키 공격에 대한 안전성 (K-KS)**

  공격자가 다른 세션키들을 알고 있어도 임시 비밀키 x, y 에 따라 세션키들의 값이 달라지기 때문에 x 와 y 값을 모르면 g^{ay}, g^{bx}, g^{xy} 를 계산할 수 없고 따라서 이전의 알려진 세션키들로부터 해당 프로토콜의 세션키를 알 수 없다.

- **전향적 보안성 (FS)**

  하나 또는 그 이상의 실체들의 비밀키 a 또는 b 가 노출되어도 정직한 실체가 생성한 이전 세션에서의 세션키를 공격자는 알 수 없다. 통신로를 통해 g^x 와 g^y 를 알게 되더라도 x 와 y 를 모르면 Diffie-Hellman 문제에 의해 g^x, g^y 로부터 g^{xy} 를 계산할 수 없기 때문이다. 따라서 g^{ay}, g^{bx} 는 알 수 있지만 $K = g^{xy+ay+bx}$ 는 계산할 수 없다.

  그러나 Unified Model 과 MQV 와 같이 FS 는 모든 세션키에 대해 EKA 가 제공될 때 확실하게 보증된다[3].

- **키 위장에 대한 안전성 (K-CI)**

  B 는 A 와 비밀 정보 K 를 공유하기 위해 $K = Y_A^y \cdot R_A^{(b+y)} (= g^{ay} \cdot g^{xb} \cdot g^{xy})$ 를 계산하게 된다. A 의 장기 비밀키 a 가 노출되었다고 가정하자. 그러나 공격자가 b 를 알지 못하면, 알 수 있는 정보 a, y, g^a, g^b, g^x, g^y 만으로는 B 와 같이 g^{xb} 을 계산할 수 없는 Diffie-Hellman 문제에 봉착하게 된다. 따라서 공격자는 A 에게 다른 실체 B 인 것처럼 위장할 수 없다.

- **미지의 키 공유에 대한 안전성 (UK-S)**

  UK-S 공격을 방지하는 방법은 첫째, CA 가 공개키 증명서를 발행할 때 각 실체가 공개키에 해당하는 비밀키를 가지고 있는지 확인하는 것이다. 그러나 최근 PKC'99 에 발표된 Blake-Wilson 의 논문[11]에서는 Duplicate Signature Key Selection(DSKS) 성질을 통해 공개키에 해당하는 비밀키를 확인하는 것만으로는 UK-S 공격을 방지하기 어렵다는 것을 밝혔다. DSKS 성질이란 주어진 서명에 대하여 그 서명에 상응하는 공개키를 여러 개 선택할 수 있다는 성질이다. Baek 과 Kim[21]은 [11]에서 제안한 키 공유 방식에 ElGamal 서명을 이용한 경우와 PKC'98 에서 Hirose 와 Yoshida 가 제안한 키 합의 프로토콜[19]이 UK-S 공격이 가능함을 제시했다. 둘째로는 키 교환 프로토콜의 메시지 인증 코드에 실체의 식별정보를 첨가하는 것이다[20]. 이것은 UK-S 공격을 방지하는 효율적인 방법 중 하나이다.

  [프로토콜 5]의 경우 3 장에서 기술한 대로 CA 는 A 가 공개키 Y_A 에 대응하는 비밀키 a 를 소유하고 있음을 검증할 수 있다고 가정했기 때문에 공격자는 A 의 공개키를 자신의 공개키인 것으로 CA 에게 증명서를 받아 B 를 속일 수 없다. 또, 생성자를 g 로 공개했기 때문에 DSKS 성질이 적용되지 않는다. 이와 같은 이유로 [프로토콜 5]은 UK-S 공격을 막을 수 있다.

2) **성능 분석**

[프로토콜 5]에서 각 실체가 계산해야 할 모듈러 지수승은 4 회이다. 실체 A 의 경우 g^x, R_B^q, Y_B^x, $R_B^{(a+x)}$를 계산하게 된다. 만약 실체 A 가 미리 알 수 있는 실체 A 의 장기키와 임시키, 실체 B 의 장기 공개키를 포함하는 값을 사전에 계산한다고 하면 온라인 상에서 실체 당 계산해야 할 모듈러 지수승은 2 회로 줄게 된다. 즉, A 의 경우 R_B^q, $R_B^{(a+x)}$ 만 계산하면 된다.

세션키로 $k = H(g^{xy+ay+bx})$ 대신에 다음과 같은 키를 사용할 수 있다.

- $k = H(g^{bx} + g^{ay} \| g^{xy})$를 사용하는 경우
  A 는 $k = H(Y_B^x + R_B^a \| R_B^x)$를 B 는 $k = H(R_A^b + Y_A^y \| R_A^y)$를 계산을 하게 된다.

- $k = H(g^{xb+ya} \| g^{xy})$를 사용하는 경우
  A 는 $k = H(Y_B^x \cdot R_B^a \| R_B^x)$를 B 는 $k = H(R_A^b \cdot Y_A^y \| R_A^y)$를 계산을 하게 된다.

- $k = H(g^{bx} \| g^{ay} \| g^{xy})$를 사용하는 경우
  A 는 $k = H(Y_B^x \| R_B^a \| R_B^x)$를 B 는 $k = H(R_A^b \| Y_A^y \| R_A^y)$를 계산을 하게 된다.

그러나 위의 세션키들을 사용할 경우 각 실체가 필요로 하는 모듈러 지수승은 5 회이며 사전 계산을 하는 경우는 3 회 연산해야 하므로 성능면에서는 떨어진다.

나. AKC 프로토콜

[프로토콜 5]는 [프로토콜 4]에서와 같이 MAC 을 추가함으로 AKC 프로토콜로 확장할 수 있다. 다음과 같이 세 개의 메시지로 구성된다.

- **[프로토콜 6]**

① A 는 $x \in_R [1, q-1]$를 선택하고 B 에게 $R_A = g^x$ 와 $Cert_A$를 보낸다.

② (a) B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod p$ 인지를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다.

(b) B 는 $y \in_R [1, q-1]$를 선택하고

$R_B = g^y$, $k' = H_1(Y_A{}^y \cdot R_A{}^{(b+y)})$, $k = H_2(Y_A{}^y \cdot R_A{}^{(b+y)})$ 와

$m_B = MAC_{k'}(2, B, A, R_B, R_A)$를 계산한다.

(c) B 는 A 에게 R_B, $Cert_B$ 와 m_B를 보낸다.

③ (a) A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod p$ 인지를 검증한다.

만약 검사가 실패하면, A 는 프로토콜을 실패로 종료한다.

(b) A 는 $k' = H_1(Y_B{}^x \cdot R_B{}^{(a+x)})$ 와 $m_B' = MAC_{k'}(2, B, A, R_B, R_A)$ 를 계산하고

$m_B' = m_B$ 인지를 검증한다.

(c) A 는 $m_A = MAC_{k'}(3, A, B, R_A, R_B)$ 와 $k = H_2(Y_B{}^x \cdot R_B{}^{(a+x)})$ 를 계산하고

B 에게 m_A 를 보낸다.

④ B 는 $m_A' = MAC_{k'}(3, A, B, R_A, R_B)$ 를 계산하고 $m_A' = m_A$ 인지를 검증한다.

⑤ 세션키는 k 이다.

1) 안전성 분석

[프로토콜 5]가 만족하는 안전성에 더하여 FS 를 확실하게 제공하고 키 확인 기능 추가로 상호 EKA 를 제공한다. 즉, IKA, EKA 와 K-KS, FS, K-CI, UK-S 에 대한 안전성을 모두 만족하고 있다.

2) 성능 분석

[프로토콜 5]에 비하여 MAC 를 계산하는 양이 추가되지만 MAC 은 효과적으로 계산될 수 있기 때문에 계산량에 큰 영향을 주지 않는다. 그러나 메시지 수는 3 회로 증가한다.

다. 2-pass 일방향 AKC 프로토콜

실제로 키 합의 프로토콜은 앞으로 이루어질 암호 통신을 위해 키를 안전하게 교환하고자 하는 프로토콜이다. 통신을 원하는 실체가 프로토콜 개시자(Initiator)가 되어 먼저 자신의 비밀 정보를 보내고 상대방의 비밀 정보를 받아 공유키를 생성한 뒤 자신이 생성한 키를 상대방도 가지고 있음을 확인하면 된다. 이와 같은 키 합의 프로토콜은 2 회의 메시지 교환으로 가능하다. 여기서는 2-pass 로 상호 실체 인증과 여러 안전성 요구사항을 제공하는 프로토콜을 설계한다.

- **[프로토콜 7]**

① A 는 $x \in_R [1, q-1]$ 를 선택하고 B 에게 $R_A = g^x$ 와 $Cert_A$, $S_A(R_A, A)$ 를 보낸다.

② (a) B 는 $1 < R_A < p$ 이고 $(R_A)^q \equiv 1 \pmod{p}$ 인지를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다.

(b) B 는 A 의 장기 공개키 Y_A 를 이용해서 S_A 를 검증한다.

만약 검사가 실패하면, B 는 프로토콜을 실패로 종료한다.

(c) $y \in_R [1, q-1]$ 를 선택하고

$R_B = g^y$, $k' = H_1(Y_A{}^y \cdot R_A{}^{(b+y)})$, $k = H_2(Y_A{}^y \cdot R_A{}^{(b+y)})$ 와

$m_B = MAC_{k'}(B, A, R_B, R_A)$ 를 계산한다.

(d) B 는 A 에게 R_B, $Cert_B$ 와 m_B 를 보낸다.

③ (a) A 는 $1 < R_B < p$ 이고 $(R_B)^q \equiv 1 \pmod{p}$ 인지를 검증한다.

만약 검사가 실패하면, A 는 프로토콜을 실패로 종료한다.

(b) A 는 $k' = H_1(Y_B{}^x \cdot R_B{}^{(a+x)})$, $k = H_2(Y_B{}^x \cdot R_B{}^{(a+x)})$ 를 계산하고

$m_B' = MAC_{k'}(B, A, R_B, R_A)$ 를 구해서 $m_B' = m_B$ 인지를 검증한다.

④ 세션키는 k 이다.

A 는 B 에게 임시 공개키를 전송하고 이에 대한 서명값을 함께 전송해서 B 에게 A 에 대한 실체 인증을 제공한다. B 는 A, B 의 ID 와 임시 비밀키를 포함하는 MAC 을 계산하고 자신의 임시 비밀키와 함께 A 에게 전송한다. A 는 B 가 보낸 임시 공개키를 이용하여 공유키를 생성하고 MAC 을 검증함으로 B 에 대한 실체 인증과 B 가 키를 가지고 있음을 확인할 수 있다. 키 확립 프로토콜 이후 암호 통신을 할 때 A 는 B 와 공유한 세션키로 암호문을 전송하게되고 B 는 A 가 보낸 암호문이 제대로 복호되는지를 확인함으로써 A 에 대한 키 확인을 하게 된다. 만약 의미없는 복호문을 얻게 된다면 프로토콜을 종료하면 된다. 이와 같이 B 에 대한 A 의 키 확인은 추후 암호 통신에서 자연스럽게 이루어지기 때문에 키 확립 프로토콜에서 생략해도 무관하다.

1) 안전성 분석

[프로토콜 7]은 앞장에서 기술한 기본적인 안전성으로 상호 IKA 를 제공하고 프로토콜 개시자에 대하여 EKA 를 제공한다. 또, 추가적 안전성 요구사항을 모두 만족하고 상호 실체 인증을 제공한다.

2) **성능 분석**

서명을 추가적으로 계산해야 하지만 부하가 가장 많은 메시지 수를 줄일 수 있으므로 [프로토콜 6]보다 성능면에서 효율적이다. 또, 임시 공개키와 서명값은 사전 계산이 가능하므로 온라인 상에서는 서명을 검증하는 계산량만 증가된다고 볼 수 있다.

5. 키 합의 프로토콜의 비교

이 장에서는 지금까지 제시한 키 합의 프로토콜을 비교한다.

<표 1>은 [프로토콜 1]~[프로토콜 7]이 공유하게 되는 비밀 정보와 이를 얻기 위해 실체 A 가 계산해야 하는 연산을 나타낸 것이다. [프로토콜 1]은 g^{ay}, g^{bx} 를 더해서 공유키를 생성하고 [프로토콜 2, 4]는 g^{ab}, g^{xy} 를 연접함으로 공유키를 생성하며, [프로토콜 3]은

$g^{xy}, g^{xb\overline{g^y}}, g^{ya\overline{g^x}}, g^{ab\overline{g^x}\,\overline{g^y}}$ 의 곱으로 공유키가 구성되어 있고 제안한 [프로토콜 5,6,7]은 g^{ay}, g^{bx}, g^{xy} 의 곱으로 구성된 공유키를 갖는다.

<표 1> 키 합의 프로토콜이 공유하는 비밀 정보

프로토콜 종류	공유 비밀 정보	실체 A 의 연산
프로토콜 1 (KEA)	$K = g^{ay} + g^{bx}$	$K = (Y_B)^x + (R_B)^a$
프로토콜 2, 4 (Unified Model)	$K = g^{ab} \parallel g^{xy}$	$K = (Y_B)^a \parallel (R_B)^x$
프로토콜 3 (MQV)	$K = g^{(x+a\overline{g^x})(y+b\overline{g^y})}$ $= g^{xy} \cdot g^{xb\overline{g^y}} \cdot g^{ya\overline{g^x}} \cdot g^{ab\overline{g^x}\,\overline{g^y}}$	$\overline{R_A} = R_A \bmod 2^{80} + 2^{80}$ $s_A = (x + a\overline{R_A}) \bmod q$ $K = (R_B (Y_B)^{\overline{R_B}})^{s_A}$
프로토콜 5,6,7 (제안 프로토콜)	$K = g^{ay+bx+xy}$ $= g^{ay} \cdot g^{bx} \cdot g^{xy}$	$K = (Y_B)^x \cdot (R_B)^{(a+x)}$

<표 2>는 두 실체가 정직하고 항상 프로토콜을 바르게 실행한다고 간주했을 때, 앞 장에서 논의한 AK, AKC 프로토콜들이 제공한다고 보는 안전성 요구사항을 나타내고 있다.

<표 2> 키 합의 프로토콜이 제공하는 안전성 요구사항

	IKA	EKA	K-KS	FS	K-CI	UK-S
프로토콜 1 (KEA)	√√	×	√√	×	√√	√√
3-pass KEA (AKC)	√√	√√	√√	×	√√	√√
프로토콜 2 (UM)	√√	×	√?	√?	×	√√
프로토콜 4 (AKC)	√√	√√	√√	√√	×	√√
프로토콜 3 (MQV)	√√	×	√√	√?	√√	×
3-pass MQV (AKC)	√√	√√	√√	√√	√√	√√
프로토콜 5 (제안한 AK)	√√	×	√√	√?	√√	√√
프로토콜 6 (제안한 AKC)	√√	√√	√√	√√	√√	√√
프로토콜 7 (제안)	√√	√I	√√	√√	√√	√√

√√ : 실체가 프로토콜을 개시하든지 그렇지 않든지 관계없이 확실성을 제공한다.

√I : 프로토콜의 개시자에게 확실성을 제공한다.

√? : 특별한 조건에 따라 확실성을 제공한다[3].

(FS 의 경우는 모든 세션키에 대해 EKA 가 가능하다면 확실성이 보증된다.)

× : 이 프로토콜에서 확실성이 제공되지 않는다.

다음 <표 3>은 키 합의 프로토콜을 계산할 때 요구되는 모듈러 지수승의 횟수를 비교한 것이다. [프로토콜 1,2,4,5,6]은 필요한 모듈러 지수승이 4 회이지만 미리 알 수 있는 자신의 장기 비밀키와 임시 공개키 그리고 상대방의 장기 공개키 값을 포함하는 양을 사전에 계산한다면 2 회로 줄어들게 된다. [프로토콜 3]의 경우는 필요한 모듈러 지수승은 3.5 회이고 사전 계산을 한 경우는 2.5 회가 된다.

앞에서 언급한 대로 MAC 은 효과적으로 계산될 수 있다. 따라서 AKC 변형들은 본질적으로 AK 와 같은 계산 비용이 든다고 볼 수 있다. 그러나 여분의 메시지 수가 필요하다. 이에 반해 [프로토콜 7]은 서명에 필요한 계산량이 추가되기는 하지만 AK 프로토콜과 같이 2 회의 메시지 수를 유지하면서 일방향 키 확인 기능을 제공하는 것을 제외하고 AKC 와 같은 안전성 요구사항을 만족한다. 사전 계산을 하는 경우에는 서명값은 미리 계산할 수 있기 때문에 실질적으로 온라인 상에서 추가되는 계산량은 서명을 검증할 때 필요한 양이다.

<표 3> 프로토콜 계산시 필요한 모듈러 지수승의 횟수

모듈러 지수승	요구되는 총수	사전계산 한 경우
프로토콜 1 (KEA)	4	2
프로토콜 2, 4 (Unified Model)	4	2
프로토콜 3 (MQV)	3.5	2.5
프로토콜 5,6 (제안 프로토콜)	4	2

<표 2>에서 비교한 대로 [프로토콜 1]은 FS 를 만족하지 않고 [프로토콜 2]는 K-CI 을 만족하지 않고 [프로토콜 3]는 UK-S 를 만족하지 않지만 [프로토콜 5]는 다른 프로토콜보다 많은 안전성을 제공한다. AK 에 MAC 를 제공하여 AKC 프로토콜로 확장하면 EKA 를 제공하게 된다. 그러나 AKC 인 3-pass KEA 는 여전히 FS 를 만족하지 않고, [프로토콜 4]는 K-CI 를 만족하지 않는다. 반면 3-pass MQV 는 [프로토콜 6]과 같이 위에서 기술한 모든 안전성 요구사항을 만족하게 된다. 그러나 성능면에서 <표 3>에서 제시한 바와 같이 사전 계산을 하게 되는 경우 온라인 상에서 요구되는 모듈러 지수승의 계산량이 [프로토콜 6]은 2 회이고 3-pass MQV 는 2.5 회로 [프로토콜 6]이 보다 적은 계산량을 필요로 한다. 또, <표 1>에서 알 수 있듯이 [프로토콜 6]은 각 실체가 공유 비밀 정보를 계산하는 과정이 MQV 보다 훨씬 단순하다.

[프로토콜 7]은 효율성을 위해서 [프로토콜 6]을 변형한 것으로 EKA 를 일방향으로만 제공하지만 실체 프로토콜 상에서는 키 확립 이후 암호 통신을 통해 상호 EKA 가 가능하기 때문에 안전성에 있어 큰 차이가 없고 성능면에 있어서는 비록 서명을 계산하기 위한 계산량이 증가하지만 부하가 훨씬 큰 메시지 수를 줄일 수 있기 때문에 보다 효율적이라고 할 수 있다.

6. 결론

본 논문에서는 IKA, EKA 와 K-KS, FS, K-CI, UK-S 에 대한 안전성을 고려하였을 때 지금까지 표준안에서 제안되어 온 다른 AK 프로토콜들보다 많은 안전성 요구사항을 제공하도록 AK 프로토콜을 제안하였다. 다음으로 MAC 을 AK 프로토콜에 추가하여 앞에서 논의한 모든 안전성 요구사항을 만족하는 AKC 프로토콜을 생성하였다. KEA 와 Unified Model 은 AKC 프로토콜로 확장하여도 여전히 만족하지 못하는 안전성 요구사항이 있는 반면 MQV 프로토콜은 제안한 AKC 프로토콜을 같이 모든 안전성 요구사항을 만족한다. 이들을 성능면에서 서로 비교해보면 사전 계산을 할 경우 제안한 프로토콜이 AKC 인 MQV 보다 온라인 상에서 계산량이 더 적게 들고 공유키를 생성하는 과정도 보다 단순함을 표를 통해 살펴보았다. 또, 이 AKC 프로토콜을 변형하여 상호 인증이 가능하고 요구되는 안전성들을 만족하는 일방향 AKC 프로토콜을 2-pass 로 설계하였다. 이는 서명값을 계산하는 계산량이 더 추가되지만

AKC 보다 메시지 수가 줄기 때문에 성능면에 있어서 보안된 프로토콜이다. 사전 계산을 하는 경우는 서명값을 미리 계산할 수 있기 때문에 온라인 상에서는 서명을 검증하는 계산량만 추가하면 된다.

이 논문에서 제안한 모든 프로토콜들은 유한체 상의 곱셈군에서 제시하였으나 타원곡선 상에서 이산 대수 문제의 어려움에 근거를 둔 방식으로 변형하여 효율성을 쉽게 향상시킬 수 있다. 여기서는 제안한 프로토콜들에 대한 안전성 요구사항을 비형식적인 방법으로 경험적 논증에 의해 분석해보았는데 B-R 과 같은 분산 환경에서 형식적으로 증명 가능한 안전성임을 보이는 것이 추후 연구 과제이다.

참고문헌

[1] S. Blake-Wilson, C. Johnson, A. Menezes, "Key Agreement Protocols and their Security Analysis", Proceedings of the sixth IMA International Conference on Cryptography and Coding, LNCS **1355**, p30-45, 1997.

[2] L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An Efficient Protocol for Authenticated Key Agreement Protocol", Technical report CORR 98-05, University of Waterloo, Canada, March 1998.

[3] S. Blake-Wilson, A. Menezes, "Authenticated Diffie-Hellman Key Agreement Protocols", Proceedings of the 5th Annual Workshop on Selected Areas in Cryptography (SAC '98), Lecture Notes in Computer Science, 1556, p339-361, 1999.

[4] ANSI X9.42, *Agreement of Symmetric Algorithm Keys using Diffie-Hellman,* working draft, May 1998.

[5] ANSI X9.63, *Elliptic Curve Key Agreement and Key Transport Protocols*, working draft, July 1998.

[6] IEEE P1363, *Standard Specifications for Public-Key Cryptography*, working draft, July 1998.

[7] ISO/IEC 11770-3, *Information Technology - Security Techniques - Key Management - Part 3: Mechanisms Using Asymmetric Techniques*, draft, 1996.

[8] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997

[9] R. Ankney, D. Hohnson and M. Matyas, "The Unified Model", contribution to X9F1, October 1995.

[10] National Security Agency, "SKIPJACK and KEA Algorithm Specification", Version 2.0, May 29, 1998.

[11] S. Blake-Wilson, A. Menezes, "Unknown Key-Share Attacks on the Station-To-Station (STS) Protocol", Technical report CORR 98-42, University of Waterloo, 1998.

[12] B. Kaliski, Contribution to ANSI X9F1 and IEEE P1363 working groups, June 1998.

[13] D. Johnson, Contribution to ANSI X9F1 working groups, June 1997.

[14] K. C. Goss, "Cryptographic Method and Apparatus for Public Key Exchange with Authentication", U.S. patent 4,956,865, September 11 1990.

[15] T. Matsumoto, Y. Takashima and H. Imai, "On Seeking Smart Public-Key Distribution Systems", The Transactions of the IECE of Japan, E69, p99-106, 1986.

[16] C. Mitchell, "Limitations of Challenge-Response Entity Authentication", Electronics Letters, **25**, p1195-1198, August 17, 1989.

[17] W. Diffie, M. E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, **22**, p644-654, 1976.

[18] M. Bellare, P. Rogaway, “Entity Authentication and Key Distributions - the Three Party Case”, Advances in Cryptology-Crypto ’93, LNCS **773**, p232-249, 1994.

[19] S. Hirose and S. Yoshida, “An Authenticated Diffie-Hellman Key Agreement Protocol Secure against Active Attacks”, Proceedings of PKC'98, Springer-Verlag, LNCS **1431**, p117-134, 1998.

[20] C. Mitchell and A. Thomas, “Standardizing Authentication Protocols Based on Public Key Techniques”, Journal of Computer Security, p23-46, 1993.

[21] J. Baek, K. Kim, “Remarks on the Unknown Key- Share Attacks”, to appear Transaction of IEICE, 2000